Fachhochschule Köln
Cologne University of Applied Sciences

Institut für Medien- und Phototechnik

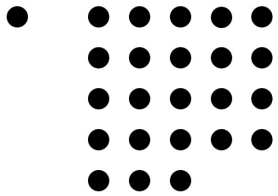
Bachelorarbeit Medientechnik

Schätzung der Brennweite mit Hilfe der Hough-Transformation

vorgelegt von
Leonie Friederike Kirk
Mat.-Nr. 11056934

Erstgutachter: Prof. Dr. rer. nat. Dietmar Kunz (Fachhochschule Köln)
Zweitgutachter: Prof. Dr.-Ing. Gregor Fischer (Fachhochschule Köln)

April 2011



Fachhochschule Köln
Cologne University of Applied Sciences

Institut für Medien- und Phototechnik

Bachelor Thesis

Focal length estimation using the Hough Transform

submitted by
Leonie Friederike Kirk
Mat.-Nr. 11056934

First Reviewer: Prof. Dr. rer. nat. Dietmar Kunz (Cologne University of Applied Sciences)
Second Reviewer: Prof. Dr.-Ing. Gregor Fischer (Cologne University of Applied Sciences)

April 2011

Bachelorarbeit

Titel: Schätzung der Brennweite mit Hilfe der Hough-Transformation

Gutachter:

- Prof. Dr. rer. nat. Dietmar Kunz (Fachhochschule Köln)
- Prof. Dr.-Ing. Gregor Fischer (Fachhochschule Köln)

Zusammenfassung: Im Rahmen dieser Bachelorarbeit wurde erfolgreich erarbeitet, dass mit Hilfe der Hough-Transformation aus Bildern der realen Welt die inneren Kameraparameter optisches Zentrum und Brennweite mit hinreichender Annäherung automatisch bestimmt werden können.

Umgesetzt wurde der Algorithmus in Java als Plug-In für das Open Source Bildverarbeitungsprogramm ImageJ. Zur Bestimmung der Fluchtpunkte wird die Hough-Transformation innerhalb des Plug-Ins zweimal durchgeführt.

Es eignet sich für Aufnahmen von Gebäuden und ähnlichen rechtwinkligen Strukturen.

Stichwörter: kaskadierte Hough-Transformation, Fluchtpunktdetektion, Brennweite, ImageJ

Datum: 18. April 2011

Bachelor Thesis

Title: Focal length estimation using the Hough Transform

Reviewers:

- Prof. Dr. rer. nat. Dietmar Kunz (Cologne University of Applied Sciences)
- Prof. Dr.-Ing. Gregor Fischer (Cologne University of Applied Sciences)

Abstract: Within the scope of this bachelor thesis it has been successfully elaborated that the intrinsic camera parameters focal length and principal point can be determined automatically with adequate approximation by using the Hough Transform.

The algorithm has been realised in Java as a plug-in for the public domain image processing program ImageJ. The Hough Transform is executed twice within this plug-in for detecting the vanishing points.

It is suited for pictures of buildings and similar orthogonal structures.

Keywords: Cascaded Hough Transform, vanishing point detection, focal length, ImageJ

Date: 18 April 2011

Inhaltsverzeichnis

1 Einleitung

In der realen Welt kommen häufig parallel und orthogonal zueinander verlaufende Geraden vor. Diese werden in fotografischen Aufnahmen perspektivisch verzerrt. Aus der Lage der Geraden im Bild lassen sich Rückschlüsse auf die internen Kameraparameter ziehen.
Durch die perspektivische Verzerrung schneiden sich die in der realen Welt parallelen Geraden in der Aufnahme jeweils in einem Fluchtpunkt. Sind die Fluchtpunkte einer Aufnahme bestimmt, lässt sich mit ihrer Hilfe das optische Zentrum und die Brennweite berechnen.
Im Rahmen dieser Bachelorarbeit sollte untersucht werden, in wie weit diese Parameter automatisch bestimmt werden können. Ziel ist ein funktionierendes Java-Plug-In für das Open Source Bildverarbeitungsprogramm ImageJ.
Eine zentrale Rolle kommt dabei der Hough-Transformation zu. Mit ihrer Hilfe können Geraden in einem Bild erkannt und ihre Schnittpunkte bestimmt werden.

2 Theoretischer Hintergrund

2.1 Fluchtpunkt

Als Fluchtpunkte eines Bildes werden die Punkte bezeichnet, in denen sich in der realen Welt parallel verlaufende Geraden durch perspektivische Verzerrung bedingt schneiden. Oft liegen diese Schnittpunkte weit außerhalb der Bildfläche.
Parallele Flächen schneiden sich in einer Fluchtgeraden. Bei waagerechten Ebenen nennt sich diese Schnittgerade Horizont. Er entspricht der Geraden, die durch die beiden horizontalen Fluchtpunkte verläuft (siehe Abb. 2.1).

Mit Hilfe des in dieser Bachelorarbeit vorgestellten ImageJ-Plug-Ins können Aufnahmen ausgewertet werden, die alle drei Fluchtrichtungen aufweisen. Bei diesen Aufnahmen ist die Bildebene in der Kamera zu keiner Geraden bzw. Kante in der realen Welt parallel. Dies führt bei der Aufnahme von Gebäuden meist zu einer Froschperspektive.

Abbildung 2.1
Aufnahme eines Gebäudes mit drei Fluchtrichtungen (rot). Der Horizont (grün) ist nach unten verschoben. Es handelt sich um eine Froschperspektive

2.2 Brennweite und optisches Zentrum

Als Brennweite wird in der Optik der Abstand des Fokuspunktes zur Hauptebene einer Linse bzw. eines Linsensystems bezeichnet [2]. In diesem Fall entspricht dies dem Abstand des Projektionszentrums der Kamera zur Bildebene. Der Lotfußpunkt des Projektionszentrums auf die Bildebene ist der Hauptpunkt des Systems. Dieser wird auch als optisches Zentrum bezeichnet und entspricht dem Höhenschnittpunkt des aus den drei Fluchtpunkten gebildeten Fluchtpunktdreiecks.
Sind die Koordinaten der Fluchtpunkte bekannt, lässt sich mit ihrer Hilfe der Höhenschnittpunkt und darüber die Brennweite berechnen.

Abbildung 2.2
Von links nach rechts 8mm, 12mm und 16mm.
Drei Aufnahmen eines Gebäudes, aufgenommen mit unterschiedlicher Brennweite. Wie in der Abbildung zu erkennen, verstärkt sich der Effekt der perspektivischen Verzerrung mit abnehmender Brennweite

2.2.1 Berechnung der Brennweite mit Hilfe der Fluchtpunkte

Seien $A=(x_a|y_a)$, $B=(x_b|y_b)$ und $C=(x_c|y_c)$ die für eine Abbildung detektierten Fluchtpunkte, lässt sich zunächst ihr Höhenschnittpunkt berechnen, welcher dem Hauptpunkt $H=(x_h|y_h)$ der Abbildung entspricht [5].

Zur Berechnung des Höhenschnittpunkts werden die Geradengleichungen zweier Höhen des Fluchtpunktdreieckes aufgestellt und diese gleichgesetzt. Da sich alle drei Höhen in einem Punkt schneiden, können dafür zwei beliebige Höhen gewählt werden. Für die Koordinaten des Hauptpunktes ergibt sich

$$x_h = \frac{y_a - y_b - x_a \cdot \frac{-x_c + x_b}{y_c - y_b} + x_b \cdot \frac{-x_c + x_a}{y_c - y_a}}{\frac{-x_c + x_a}{y_c - y_a} - \frac{-x_c + x_b}{y_c - y_b}} \quad \text{und} \quad y_h = x_h \cdot \frac{-x_c + x_a}{y_c - y_a} + y_b - x_b \cdot \frac{-x_c + x_a}{y_c - y_a}.$$

Der Abstand des Projektionszentrums Z der Kamera zur Abbildung entspricht der Strecke $\overline{HZ}$. Diese ist die effektive Brennweite f des Systems.

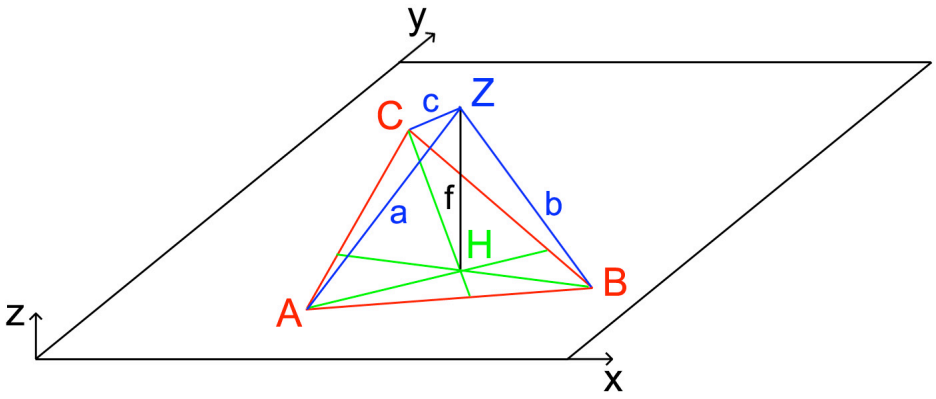


Abbildung 2.2.1
Fluchtpunktdreieck ABC in der Bildebene mit Hauptpunkt H und Projektionszentrum der Kamera Z

Die Vektoren $\vec{a}$, $\vec{b}$ und $\vec{c}$ (siehe Abb. 2.2.1) stehen senkrecht aufeinander, da die Geraden in der realen Welt, die sich in den Fluchtpunkten A, B und C schneiden, senkrecht aufeinander stehen. Daher ergibt ihr Skalarprodukt jeweils 0.
Sie lassen sich beschreiben als

$$\vec{a} = (x_h - x_a, y_h - y_a, -f)^T,\ \vec{b} = (x_h - x_b, y_h - y_b, -f)^T \text{ und } \vec{c} = (x_h - x_c, y_h - y_c, -f)^T.$$

Mit Hilfe des Hauptpunktes und zwei Fluchtpunkten lässt sich jetzt die Brennweite berechnen.

$$< a,b >= 0$$
$$< a,b >= (x_h - x_a)(x_h - x_b) + (y_h - y_a)(y_h - y_b) + (-f_{ab})^2 = 0$$
$$f_{ab}^2 = -((x_h - x_a)(x_h - x_b) + (y_h - y_a)(y_h - y_b))$$
$$f_{ab} = \sqrt{-((x_h - x_a)(x_h - x_b) + (y_h - y_a)(y_h - y_b))}$$

Analog dazu gilt für f_{bc} und f_{ac}

$$f_{ac} = \sqrt{-((x_h - x_a)(x_h - x_c) + (y_h - y_a)(y_h - y_c))}$$
$$f_{bc} = \sqrt{-((x_h - x_b)(x_h - x_c) + (y_h - y_b)(y_h - y_c))}$$

sowie

$$f = f_{ab} = f_{ac} = f_{bc}.$$

3 Hough-Transformation

Die Hough-Transformation ist ein robustes globales Verfahren zur Erkennung von Geraden und anderen parametrisierbaren geometrischen Formen in Bildern. Sie wurde von P. V. C. Hough im Jahre 1962 unter dem Namen „Method and means for recognizing complex patterns“ patentiert [6]. Für die Detektion der Fluchtpunkte eines Bildes ist lediglich die Erkennung von Geraden mittels der Hough-Transformation im Bild interessant.

3.1 Funktionsweise der Hough-Transformation

Die originale Hough-Transformation verwendet die Parameter Steigung und Y-Achsenabschnitt zur Beschreibung einer Geraden in der Form $y = mx + n$. Die Hough-Transformation benötigt zunächst ein binäres oder Grauwert-Kantenbild des Ausgangsbildes. Alle Pixel oberhalb eines festgelegten Grauwertes (im einfachsten Falle alle weißen Pixel) werden prozessiert. Dabei werden für jeden Pixel alle Kombinationen aus Steigung und Y-Achsenabschnitt von Geraden, welche diesen Pixel beinhalten würden, im Hough-Raum um Eins hochgezählt. Es ergibt sich für einen einzelnen Punkt im Kantenbild eine Gerade im Hough-Raum. Umgekehrt entspricht ein Punkt im Hough-Raum einer Geraden im Kantenbild. Die Koordinaten der Maxima im Hough-Raum entsprechen den Parametern der im Kantenbild erkannten Geraden.
Dieser Ansatz hat sich jedoch als ungeeignet erwiesen, da die Steigung bei senkrechten Geraden unendlich ist und somit im endlichen Parameterraum nicht erfasst werden kann.

Durchgesetzt hat sich zumeist die Beschreibung einer Gerade mittels der Hesseschen Normalform. Bei dieser Variante wird die Gerade durch ihren Winkel α zur X-Achse sowie ihren euklidischen Abstand d zum Koordinatenursprung in der Mitte des Bildes beschrieben. Es ergibt sich als Parametergleichung $d = x \cdot \cos(\alpha) + y \cdot \sin(\alpha)$. Hierbei wird ein Punkt im Kantenbild auf eine sinusförmige Kurve im Hough-Bild abgebildet [1].
Endpunkte von Geraden werden von der klassischen Hough-Transformation nicht erfasst.

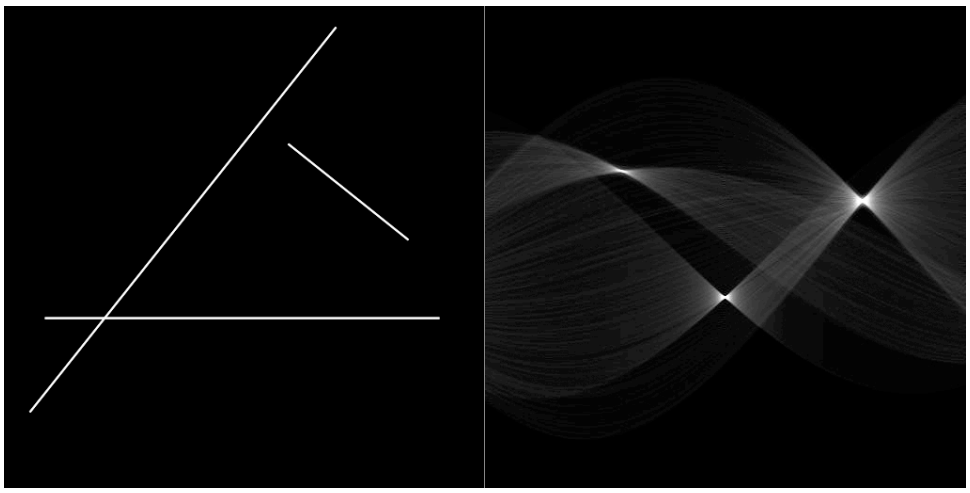


Abbildung 3.1
Links synthetisch erzeugtes Kantenbild mit drei Geraden, rechts der zugehörige Hough- bzw. Parameterraum. Gut erkennbar sind die drei Maxima im Hough-Raum, deren Koordinaten den Parametern Winkel und Abstand zur Mitte der drei Geraden entsprechen. Es wurde die Parametergleichung der Hesseschen Normalform verwendet

3.2 Die kaskadierte Hough-Transformation

Die Dualität zwischen Kantenbild und Hough-Raum bei der Parametrisierung mit Steigung und Y-Achsenabschnitt ist eine nützliche Eigenschaft zur Fluchtpunktdetektion. Um das Problem des unbeschränkten Parameterraumes zu beheben, wird der Hough-Raum in drei beschränkte Unterräume zerlegt [3].
Hierbei wird eine leicht abgewandelte Parametergleichung verwendet:

$$ax + b + y = 0$$

Das Koordinatensystem eines Bildes verläuft vom oberen, linken Bildpunkt ausgehend nach rechts sowie nach unten. Das kartesische Koordinatensystem verläuft mit Ursprung in der Mitte des Bildes ebenfalls von links nach rechts, die Y-Achse jedoch von unten nach oben (siehe Abb. 3.2a). Durch die abgewandelte Parametergleichung wird die Y-Achse automatisch horizontal gespiegelt.

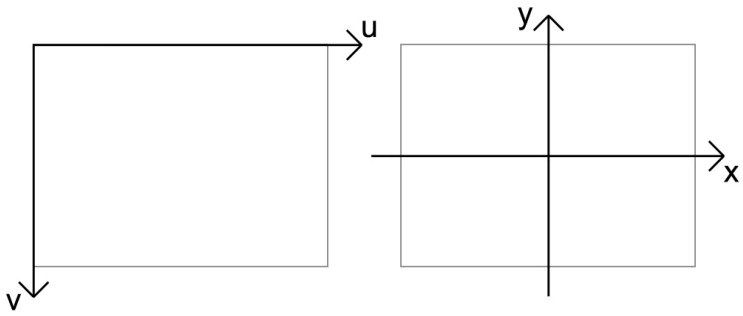


Abbildung 3.2a
Links Bildkoordinatensystem, rechts kartesisches Koordinatensystem

Das Kantenbild entspricht dem (x, y)-Raum, der Hough-Raum ist somit der (a, b)-Raum. Der (a, b)-Raum wird auf einen Wertebereich von -1 bis 1 beschränkt, $|a| \leq 1$ und $|b| \leq 1$. Ist $|a| > 1$ und $|a| \geq b$, wird der Punkt (a, b) in einen zweiten Unterraum mit den Koordinaten $1/a$ und b/a übertragen. Punkte mit Koordinaten $b > 1$ und $b > a$ werden schließlich in einen dritten Unterraum mit den Koordinaten $1/b$ und a/b abgebildet. Der unbeschränkte Hough-Raum wird in drei beschränkte Unterräume mit den Intervallen [-1|1] aufgeteilt.
Das gleiche Prinzip wird auf den (x, y)-Raum, also auf das Kantenbild angewandt. Alle $|x| \leq 1$ und $|y| \leq 1$ werden in den ersten Unterraum mit den Koordinaten x und y abgebildet, alle $|x| > 1$ und $|x| \geq y$ kommen in den zweiten Unterraum ($1/x$, y/x) und alle $y > 1$ und $y > x$ landen im dritten Unterraum mit den Koordinaten ($1/y$, x/y). Der erste Unterraum (x, y) entspricht hierbei dem Kantenbild. Handelt es sich um ein nicht quadratisches Kantenbild, füllt die längere Seite das Intervall von -1 bis 1 aus.
Damit können auch weit außerhalb des Bildes liegende Fluchtpunkte erfasst werden.

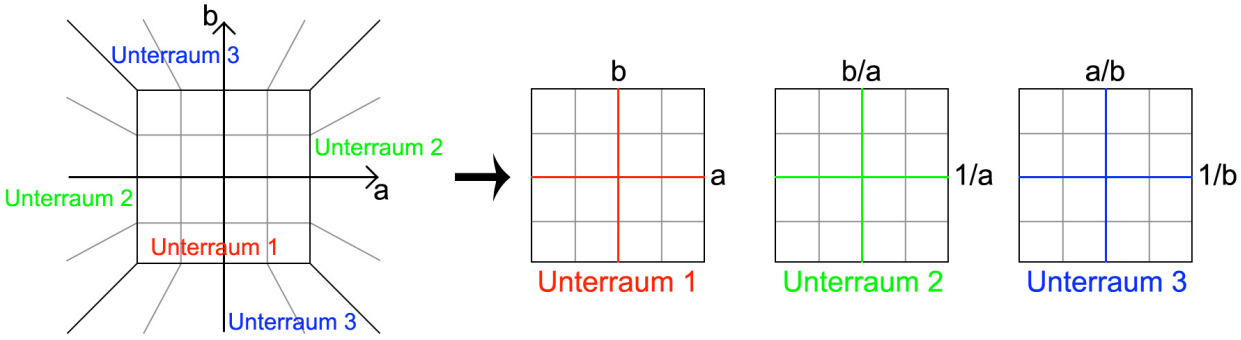


Abbildung 3.2b
Unterteilung des unbeschränkten Hough-Raumes in drei beschränkte Unterräume mit den Koordinaten (a, b), ($1/a$, b/a) und ($1/b$, a/b) [3]

3.2.1 Fluchtpunkte im Hough-Raum

Aus jedem Punkt im Kantenbild bzw. (x, y)-Raum wird eine Gerade im Hough- bzw. (a, b)-Raum und umgekehrt. Durch diese Symmetrie zwischen Kantenbild und Hough-Raum ist es möglich, mit wiederholter Hough-Transformation zwischen den beiden Räumen hin und her zu wechseln.
Wird die kaskadierte Hough-Transformation auf ein Kantenbild angewendet, ergibt dies zunächst den ungefilterten Hough-Raum. Ein geeigneter Filter extrahiert die lokalen Maxima des Hough-Raumes.

Die Koordinaten dieser Maxima entsprechen den Parametern der Geraden im Kantenbild. Wird jetzt die Hough-Transformation auf das gefilterte Ergebnis der vorangegangenen Hough-Transformation angewendet, ergibt sich im ersten Unterraum das Bild der zuvor erkannten Geraden des Kantenbildes. Im Hough-Raum liegen die Maxima transformierter Geraden, die sich im Kantenbild schneiden oder bei entsprechender Länge schneiden würden, auf einer Geraden. Eine zweite Hough-Transformation erkennt diese Gerade. Die Koordinaten der Maxima im zweiten Hough-Raum sind die Koordinaten der Schnittpunkte der Geraden des Kantenbildes. Dass die Hough-Transformation in ihrer üblichen Form keine Informationen über Start- und Endpunkte von Geraden liefert, ist für die Detektion der Fluchtpunkte nicht von Bedeutung.

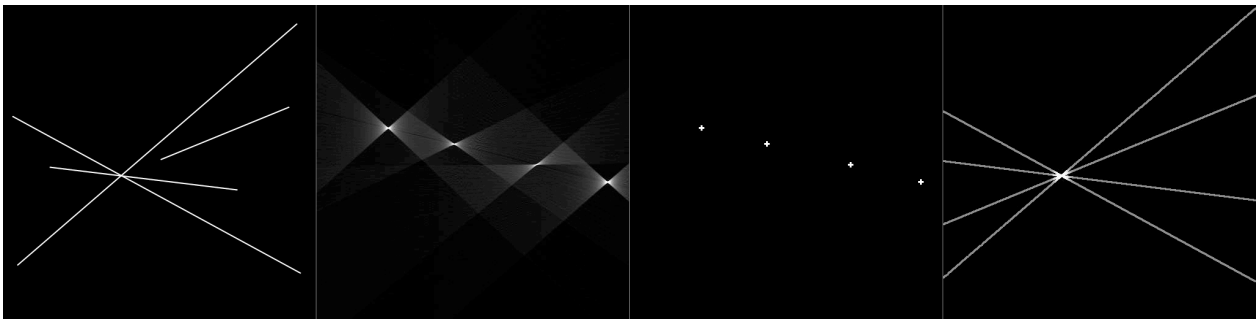


Abbildung 3.2.1a
Von links nach rechts: Synthetisches Kantenbild, zugehöriger Hough-Raum, detektierte Maxima des Hough-Raumes und das Ergebnis einer auf das Maxima-Bild angewandten Hough-Transformation. Zur besseren Erkennbarkeit sind die Maxima hier als kleine Kreuze markiert und die Linien im Ergebnisbild stärker dargestellt

Fluchtpunkte liegen meist weit außerhalb des Bildes, können also nach der zweiten Hough-Transformation als Maxima im zweiten und dritten Unterraum detektiert werden. Sind die Koordinaten möglicher Fluchtpunkte gefunden, können diese auf das kartesische Koordinatensystem des (x, y)-Raumes umgerechnet werden.

Dabei ergeben sich je nach Unterraum die folgenden Umrechnungsformeln:

Unterraum 1:
$$x = a$$
$$y = b$$

Unterraum 2:
$$x = 1/a$$
$$y = x \cdot b$$

Unterraum 3:
$$y = 1/a$$
$$x = y \cdot b$$

Mit den so berechneten Fluchtpunkten lässt sich jetzt wie unter 2.2.1 beschrieben die Brennweite annähernd berechnen.

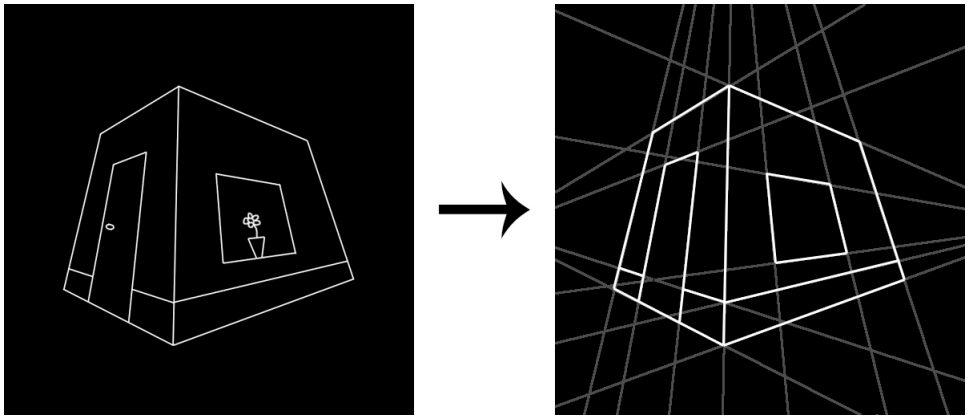


Abbildung 3.2.1b

Synthetisches Kantenbild eines kleines Häuschens und der erste Unterraum nach der zweiten Hough-Transformation. Zur besseren Erkennbarkeit sind die Maxima mit kleinen Kreuzen hervorgehoben, die Geraden des Ergebnisbildes verbreitert und das Häuschen hervorgehoben

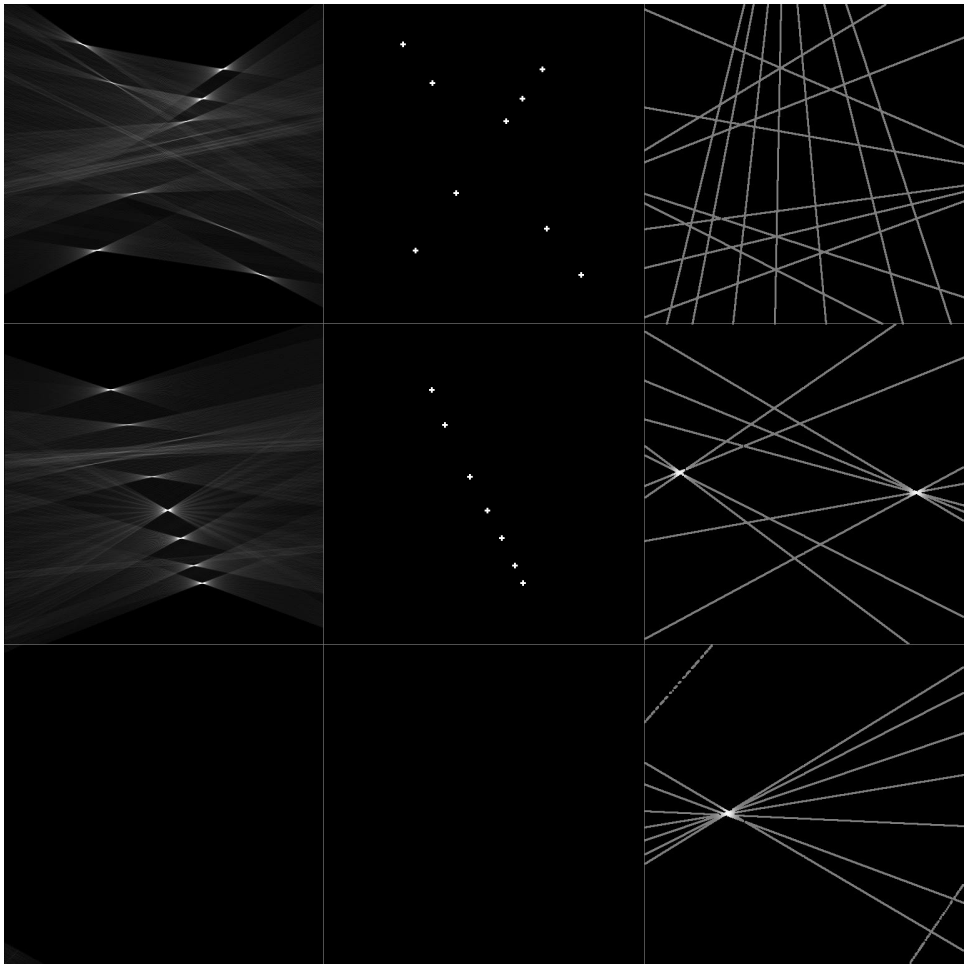


Abbildung 3.2.1c

Von oben nach unten jeweils den ersten, zweiten und dritten Unterraum. Von links nach rechts zunächst das Ergebnis der ersten Hough-Transformation, dann die detektierten Maxima und zuletzt das Ergebnis der zweiten Hough-Transformation. Sehr schön zu erkennen sind die Fluchtpunkte, die außerhalb des ersten Unterraumes, das heißt außerhalb des Bildes liegen

4 Aufbau des Programms

Bevor auf die eigentliche Implementierung des Plug-Ins eingegangen wird, werden die einzelnen Schritte zum besseren Verständnis im Vorfeld erläutert.

4.1 Vorbereitung des Bildmaterials

Damit sich die Bilder für die folgenden Hough-Transformationen eignen, sind im Vorfeld einige Schritte nötig.

4.1.1 Transformation in ein 8bit-Grauwertbild

In der Regel werden Bilder mit Digitalkameras im RGB-Farbraum aufgenommen. Zur weiteren Verarbeitung der vorliegenden Daten sind in der Folge 8bit-Grauwertbilder Voraussetzung. Daher erfolgt an erster Stelle die Umwandlung des Datensatzes in ein 8bit-Grauwertbild. Natürlich können auch direkt 8bit-Grauwertbilder genutzt werden. In diesem Fall würde dieser Schritt entfallen.

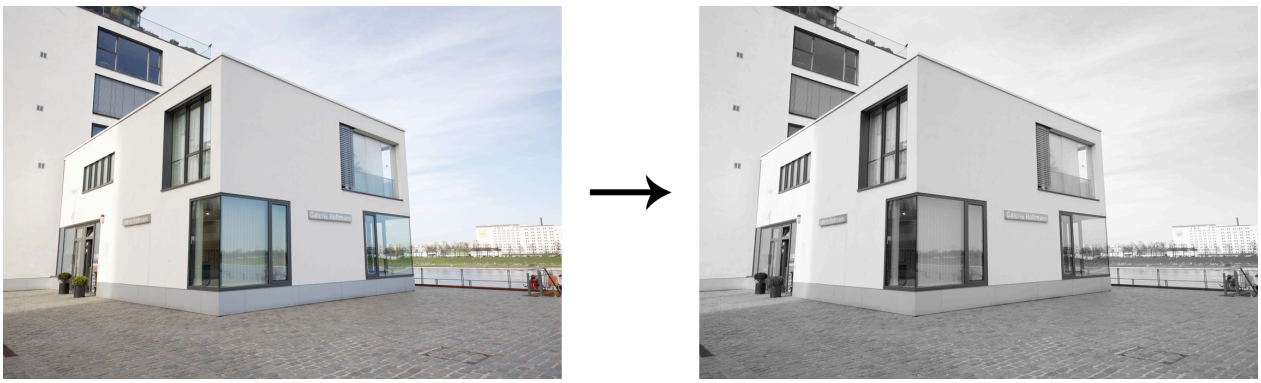


Abbildung 4.1.1
Konvertierung des Eingangsbildes in ein 8bit-Grauwertbild

4.1.2 Kantendetektor

Auf das 8bit-Grauwertbild wird im zweiten Schritt der Sobel-Operator als Kantendetektor angewendet. Die Hough-Transformation benötigt zwingend ein Kantenbild als Eingangsbild. Andernfalls würde sie keine sinnvollen Ergebnisse liefern, da jeder Pixel oberhalb einer angegebenen Grauwertgrenze prozessiert würde, beispielsweise der komplette Himmel.
Vielmehr muss selbst für das Kantenbild noch ein Schwellwert gesetzt werden kann um weniger ausgeprägte Kanten nicht mit einzubeziehen. Ziel ist es, die prägnanten Geraden des Bildes herauszustellen um möglichst optimale Eingangsbilder für die Hough-Transformation

zu erhalten. Bäume, Pflastersteine und ähnliche für eine Fluchtpunktdetektion störenden Elemente können über diesen Schwellwert größtenteils rausgefiltert werden.

Abbildung 4.1.2
Auf das Grauwertbild wird ein Kantendetektor angewendet

4.1.3 Anpassung der Arbeitsfläche

Da die einzelnen Hough-Räume quadratisch mit dem Intervall [-1, 1] angelegt sind, ist es sinnvoll, das Bild im ersten Schritt an das quadratische Format anzupassen. Hierzu wird es zentriert in ein quadratisches schwarzes Bild eingefügt, dessen Kantenlänge der doppelten Länge der längeren Bildseite des Ausgangsbildes entspricht (siehe Abb. 4.1.3). Auch wenn sich die zu verarbeitende Fläche dadurch mindestens vervierfacht, ist keine zusätzliche Rechenleistung erforderlich, da schwarze Pixel nicht verarbeitet werden. Dieses Verfahren bringt jedoch Vorteile mit sich.

Durch den schwarzen Rahmen um das eigentliche Kantenbild wird bei der ersten Hough-Transformation vermieden, dass Ergebnisse für den dritten Unterraum des Hough-Raumes berechnet werden. Kein Punkt des Bildes kann so nah am Rand liegen, dass der Y-Achsenabschnitt einer Geraden, welche diesen Punkt durchläuft, größer ist als die zugehörige Steigung (jenseits des Intervalls [-1, 1]).

Bei den für dieses Programm auswertbaren Bildern würden im dritten Unterraum keine für den weiteren Verlauf nützlichen Informationen abgelegt. Dadurch kann verhindert werden, dass dort fälschlicherweise Maxima detektiert werden. Auch liegen die Maxima der ersten beiden Unterräume durch diesen schwarzen Rahmen in ihrem Hough-Raum zentraler und können somit sicherer detektiert werden.

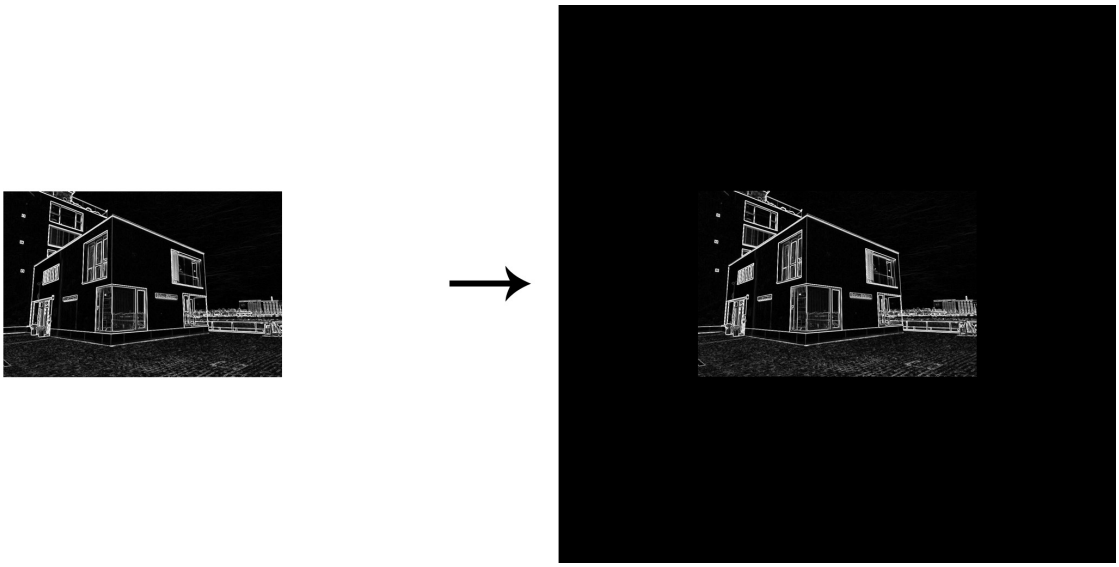


Abbildung 4.1.3
Die Arbeitsfläche wird vergrößert mit der doppelten, längeren Bildseite als neue Kantenlänge des Bildes

4.2 Die erste Hough-Transformation

Nach den Vorbereitungen des Bildes wird die erste Hough-Transformation angewendet. Das Eingangsbild ist also das quadratische, vergrößerte Kantenbild. Die Auflösung des Hough-Raumes kann im Programm eingestellt werden. Das Ergebnisbild der ersten Hough-Transformation beinhaltet die drei Unterräume, die untereinander angeordnet sind. Der dritte Unterraum bleibt zunächst schwarz.

In den ersten beiden Unterräumen sollten die drei Geraden, auf denen die Maxima der sich jeweils in einem Fluchtpunkt schneidenden Geradenscharen liegen, optisch bereits zu erkennen sein.

4.2.1 Detektion der Maxima

Anschließend gilt es, die Maxima im Hough-Bild zu detektieren. Ein geeigneter Operator ist der Harris Corner Detector. Er erkennt markante Stellen im Bild, welche sich möglichst stark von ihrer Nachbarschaft unterscheiden. Eckpunkte sind meist dort gegeben, wo der Gradient der Bildfunktion in vertikaler und horizontaler Richtung besonders hoch ist. Der Harris Corner Detector arbeitet daher mit der ersten partiellen Ableitung der Bildfunktion in beide Richtungen [1].
Die Detektion von Kanten, bei denen der Gradient nur in eine Richtung hoch ist, kann auf diese Weise vermieden werden.

Der Abstand, den die einzelnen Maxima voneinander mindestens aufweisen müssen, kann im Programm eingestellt werden. Über den Parameter α kann die Empfindlichkeit des Detektors gesteuert werden. Je größer α ist, desto weniger Eckpunkte werden gefunden. Ebenso wird ein Schwellwert für die „Ecken-Stärke“ angegeben.
Die auf diese Weise detektierten Ecken werden in absteigender Reihenfolge ihrer Ecken-Stärke in einer Liste gesammelt. In einer festgelegten räumlichen Umgebung werden alle Eckpunkte bis auf den stärksten gelöscht.

Für die Weiterverarbeitung werden die auf diese Weise detektierten Maxima als weiße Punkte in ein schwarzes Bild der Größe des Hough-Raumes eingetragen.

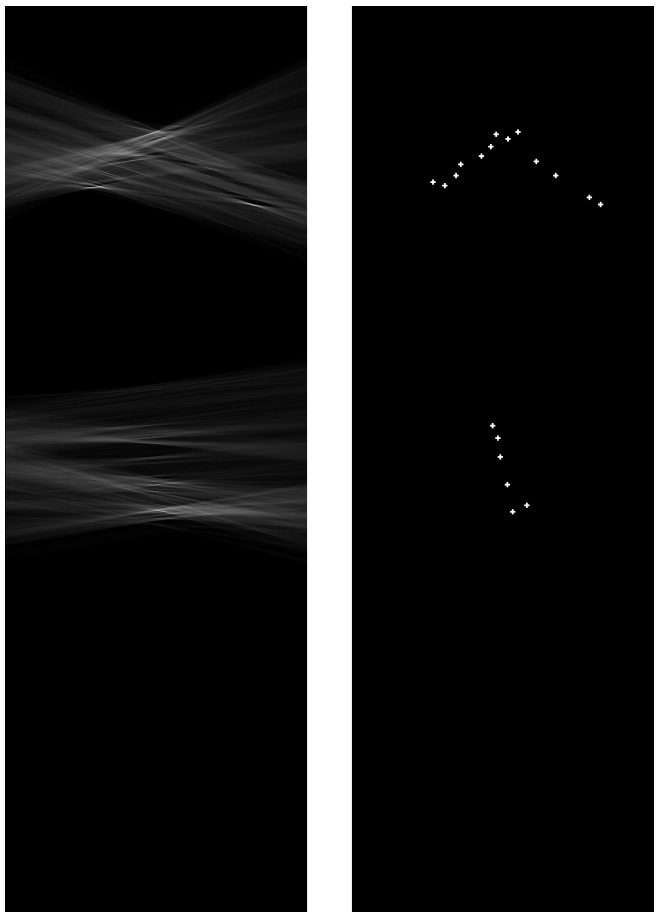


Abbildung 4.2.1
Die linke Seite der Abbildung zeigt das Ergebnis der ersten Hough-Transformation. Rechts sind die detektierten Maxima, welche zur besseren Erkennbarkeit als kleine Kreuze markiert sind. Im Plug-In selbst werden sie als weiße Punkte eingetragen

4.3 Die zweite Hough-Transformation

Die zweite Hough-Transformation wird auf das Bild der detektierten Maxima angewendet. Dabei müssen die einzelnen Unterräume unterschiedlich behandelt werden. Auf den ersten Unterraum kann zunächst die Hough-Transformation, wie sie bereits in 4.2 verwendet wird, angewendet werden. Der neue Hough-Raum, der im Grunde wieder dem Bildraum bzw. (x, y)-Raum entspricht, ist ebenfalls in die drei Unterräume unterteilt.
Die Ergebnisse der Hough-Transformation des zweiten Unterraumes werden ebenfalls in diese neuen Unterräume geschrieben. Da die Koordinaten des zweiten Unterraumes jedoch ($1/x$) und (y/x) sind, ergibt sich für einen Punkt ($1/x$, y/x) im zweiten Unterraum die folgende Geradengleichung für den ersten Unterraum des neuen Hough-Raumes (a, b)

$$a \cdot \frac{1}{x} + b + \frac{y}{x} = 0 \Leftrightarrow a + bx + y = 0 \,.$$

In den zweiten Unterraum wurden die Ergebnisse geschrieben, bei denen der Wert der Steigung größer war als der des Y-Achsenabschnittes. Der Bereich des zweiten Unterraumes, dessen Inhalte einen Y-Achsenabschnitt höher als die zugehörige Steigungen haben könnten, beinhaltet daher keine Maxima, die einer Rücktransformation in den (x, y)-Raum bedürfen.

Da der dritte Unterraum des Bildes der detektierten Maxima schwarz ist, muss dieser ebenfalls nicht verarbeitet werden.

Die zweite Hough-Transformation ist eine Rücktransformation in den Bildraum bzw. (x, y)-Raum. Der erste Unterraum entspricht nun dem Eingangsbild vor der ersten Hough-Transformation, also dem vergrößerten Kantenbild. Er enthält die durch die erste Hough-Transformation erkannten Geraden.
Bei deutlich perspektivisch verzerrten Bildern können sich diese Geraden bereits im ersten Unterraum in ihren Fluchtpunkten schneiden. Oft finden sich jedoch die beiden Fluchtpunkte in horizontaler Richtung im zweiten Unterraum wieder. Der dritte Unterraum enthält in der Regel den dritten Fluchtpunkt.

Diese drei Fluchtpunkte gilt es anschließend ebenfalls zu detektieren.

4.3.1 Detektion der Fluchtpunkte

Die Detektion der Fluchtpunkte entspricht weitestgehend der Detektion der Maxima (siehe 4.2.1). Es empfiehlt sich lediglich, die einstellbaren Parameter des Harris Corner Detektors an die Suche anzupassen. Fluchtpunkte liegen beispielsweise deutlich weiter auseinander als die in Schritt 4.2.1 zu detektierenden Maxima. Als Mindestabstand zwischen zwei „Ecken“ kann also ein deutlich größerer Wert gewählt werden. Dies verringert die Detektion falscher Fluchtpunkte.

Auch wird die Anzahl der zu findenden Eckpunkte begrenzt. Nur die stärksten Ecken werden ausgegeben. Da zur Berechnung der Brennweite im nächsten Schritt jeweils drei Fluchtpunkte benötigt werden, ist dies die vorgegebene Mindestanzahl zu detektierender „Ecken“.
Um die Chance zu vergrößern, dass die drei tatsächlichen Fluchtpunkte auch gefunden werden, kann man eine größere Anzahl potentieller Fluchtpunkte suchen lassen.
Die Fluchtpunkte werden ebenfalls als weiße Punkte in einem schwarzen Bild der Größe des Hough-Raumes abgelegt.

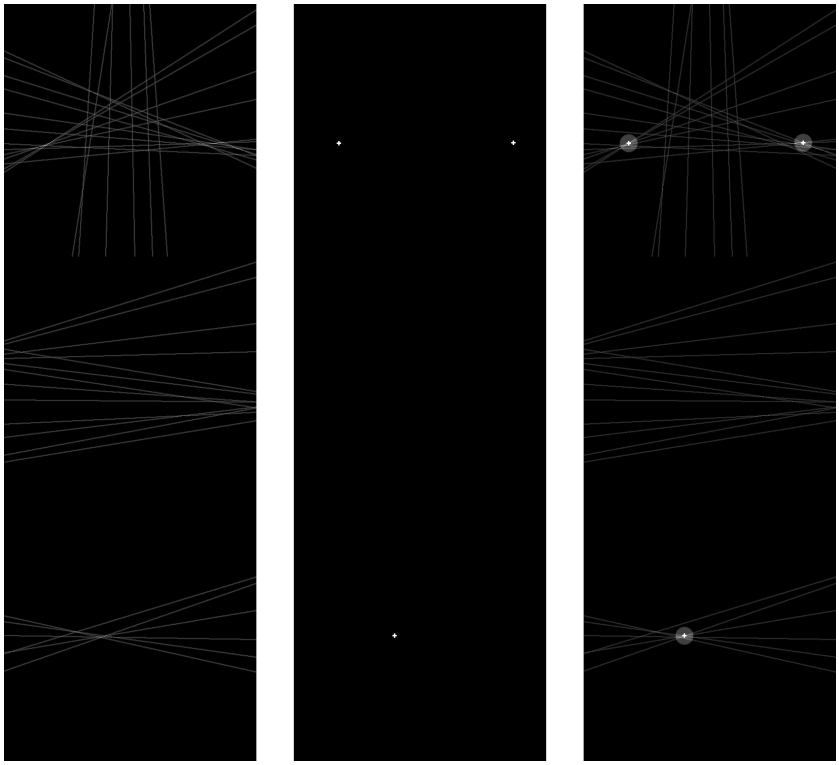


Abbildung 4.3.1
Links ist das Ergebnis der zweiten Hough-Transformation. Die Mitte zeigt drei detektierte Fluchtpunkte. Rechts sind die detektierten Fluchtpunkte im Ergebnisbild der zweiten Hough-Transformation markiert

4.4 Berechnung der Brennweite

Eine mögliche Brennweite des Bildes lässt sich aus jeweils drei potentiellen Fluchtpunkten berechnen. Wurden in Schritt 4.3.1 mehr als drei Fluchtpunkte detektiert, wird zu jedem möglichen Fluchtpunkttripel die Brennweite berechnet.
Kann die Brennweite mit einem Fluchtpunkttripel nicht berechnet werden oder liegt der zugehörige Hauptpunkt nicht in der Nähe des Bildzentrums, wird dieses automatisch aussortiert.

4.4.1 Berechnung der kartesischen Koordinaten der Fluchtpunkte

Es müssen zunächst die Bildkoordinaten der detektierten Fluchtpunkte in kartesische Koordinaten umgerechnet werden. Wenn (u, v) die Bildkoordinaten eines Punktes sind, (x, y) seine kartesischen Koordinaten und b die Bildbreite sowie h die Bildhöhe, welche der dreifachen Bildbreite entspricht, dann gilt für alle Punkte im ersten Unterraum

$$x = 2 \cdot \frac{u - \frac{w}{2}}{\frac{w}{2}} \text{ und } y = 2 \cdot \frac{v - \frac{h}{6}}{\frac{h}{6}} \cdot (-1).$$

Die Multiplikation der Koordinaten mit 2 erfolgt auf Grund der verdoppelten Kantenlänge des ursprünglichen Bildes in Schritt 4.1.3. Sie werden dadurch wieder auf die ursprüngliche Bildgröße zurückgerechnet. Die Multiplikation mit -1 bewirkt die Spiegelung in horizontaler Richtung.

Die Koordinaten des zweiten Unterraumes müssen zunächst dahingehend umgerechnet werden, als würde der zweite Unterraum oben links ebenfalls mit (0,0) beginnen. Danach werden die ($1/x$, y/x)-Koordinaten auf (x, y)-Koordinaten zurückgerechnet. Daher ist

$$x = 2 \cdot \frac{\frac{w}{2}}{u - \frac{w}{2}} \text{ und } y = x \cdot \frac{\frac{h}{6}}{v - \frac{h}{2}} \cdot (-1).$$

An dieser Stelle muss y nicht mehr mit 2 multipliziert werden, da es mit dem bereits zurückskalierten x berechnet wird.

Für die Koordinaten des dritten Unterraumes gilt

$$y = 2 \cdot \frac{\frac{w}{2}}{u - \frac{w}{2}} \cdot (-1) \text{ und } x = y \cdot \frac{\frac{h}{6}}{v - \frac{5h}{6}} \cdot (-1).$$

Diesmal werden y und x mit -1 multipliziert, da y horizontal gespiegelt wird und x mit y berechnet wird. x muss folglich zurückgespiegelt werden.

4.4.2 Berechnung des optischen Zentrums H und der Brennweite f

Aus den Koordinaten der Fluchtpunkte lassen sich die Koordinaten des optischen Zentrums bzw. Hauptpunktes berechnen. Dieser ist der Höhenschnittpunkt des Fluchtpunktdreiecks. Die Brennweite kann jetzt mit Hilfe der Fluchtpunkte und des Hauptpunktes bestimmt werden (siehe 2.2.1).

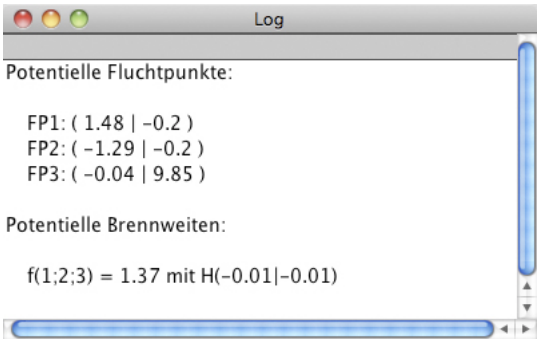


Abbildung 4.4.2
Ergebnisfenster mit Koordinaten drei potentieller Fluchtpunkte,
den Koordinaten des optischen Zentrums und mit der Brennweite

4.4.3 Berechnung der Brennweite in mm

Ist die Sensorbreite der Kamera, aus welcher das Eingangsbild stammt, bekannt, so lässt sich mit ihrer Hilfe die effektive Brennweite in mm berechnen.
Die Abmessungen des Bildes wurden zu Beginn auf das Intervall [-1, 1] skaliert. Der Wert der Eins entspricht der halben Bildbreite in Pixeln. Folglich entspricht Eins ebenfalls der halben Sensorbreite in mm.

Eine Multiplikation der berechneten Brennweite mit der halben Sensorbreite in mm ergibt die effektive Brennweite des Aufnahmesystems in mm.

Falls das Bild in eine Richtung beschnitten wurde, beispielsweise von einem 3:2-Format auf ein quadratisches Format, so kann dieser Cropfaktor mit einbezogen werden. Im Falle 3:2 auf 1:1 betrüge der Cropfaktor 1,5:

$$Cropfaktor = \frac{Seitenlänge_{alt}}{Seitenlänge_{neu}}$$

Sind Brennweite und/oder Cropfaktor bekannt, können sie im Programm für die Berechnung mit angegeben werden.

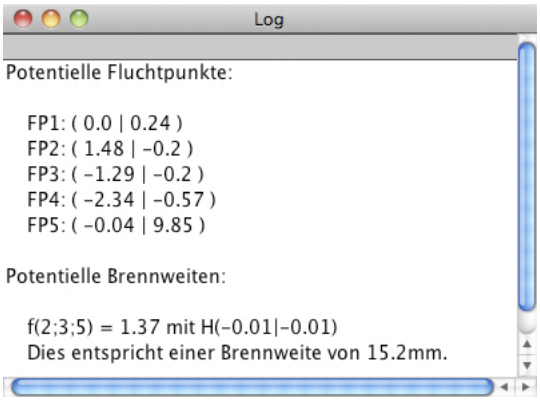


Abbildung 4.4.3
Ergebnisfenster mit fünf möglichen Fluchtpunkten. Da lediglich der Hauptpunkt eines Fluchtpunkttripels in der Nähe des Bildzentrums liegt, wird ausschließlich die zugehörige Brennweite berechnet. Mit Hilfe der Sensorbreite in mm der Kamera, aus welcher das Eingangsbild stammt, konnte die Brennweite in mm ebenfalls berechnet werden

5 Implementierung

Das Programm wurde als Plug-In für das open-source Bildverarbeitungsprogramm ImageJ [p1] realisiert. Es ist mit Hilfe von Eclipse [p2] in Java programmiert.

Der vollständige Quellcode des Plug-Ins liegt der Bachelorarbeit auf CD bei.

5.1 Die Ausgangsbasis

W. Burger und M. Burge [1] haben die Hough-Transformation bereits als ImageJ-Plug-In in Java umgesetzt. Sie haben die Hough-Transformation jedoch mit der Hesseschen Normalform implementiert (siehe 3.1). Wie bereits beschrieben ist diese Variante im hier vorliegenden Fall ungeeignet. Um nicht bei Null anzufangen, bot es sich jedoch an, diesen Quellcode als Grundgerüst zu verwenden und darauf aufzubauen.

Ebenfalls verwendet wurde ihre Version des Harris Corner Detectors für ImageJ. Dieser eignet sich in seiner Form für die Detektion der Maxima innerhalb dieses Plug-Ins. Lediglich die Ausgabe der Maxima wurde geändert. Sie werden nun als weiße Punkte in einem schwarzen Bild markiert, wohingegen sie im Original als schwarze Kreuze auf einer blasseren Version des Ausgangsbildes eingezeichnet werden.

Ein Kantendetektor in Form eines Sobel-Operators ist in ImageJ bereits implementiert und muss lediglich aufgerufen werden. Seine Methode heißt `findEdges()` und kann auf Byte-Prozessoren angewendet werden.

5.2 Die Klassen des Programms und seine Methoden

Das Plug-In besteht aus sieben Klassen.

`PluginApproximateFocalLength_` ist die Klasse des Plugins, die von ImageJ aus aufgerufen wird, nachdem das zu verarbeitende Bild geöffnet wurde. Sie ruft alle weiteren Klassen auf.

`Cascaded1HT` ist die erste Klasse, die innerhalb des Plugins aufgerufen wird. Sie berechnet die erste Hough-Transformation der kaskadierten Hough-Transformation.

`HarrisCornerDetector` detektiert darauf hin die Maxima im Hough-Raum der ersten Hough-Transformation und gibt diese zur Weiterverarbeitung geeignet aus.

`Corner` ist ein Datentyp, welchen der Harris Corner Detector benötigt.

`Cascaded2HT` wendet auf dieses Bild die zweite Hough-Transformation an. Im Gegensatz zur ersten ist dieses etwas aufwendiger, da drei Unterräume verarbeitet werden müssen.

`HCDVanishingPoints` sucht die Maxima im zweiten Hough-Raum. Diese Klasse entspricht dem `HarrisCornerDetector`, nutzt jedoch deutlich andere Parameterwerte zur Detektion der Fluchtpunkte als Ecken.

`CalculateFL` berechnet schließlich die Koordinaten der Fluchtpunkte und des Hauptpunktes sowie die effektive Brennweite.

Im Folgenden werden fünf Methoden innerhalb des Programmcodes näher erläutert. Sie sind von zentraler Bedeutung und sollten daher hier Erwähnung finden.

5.2.1 Die Methode doPixel( )

Die Methode `doPixel()` ist Bestandteil der Klasse `Cascaded1HT` und wird auf jeden Pixel angewendet, dessen Grauwert oberhalb eines Schwellwertes liegt.
Zu einem entsprechenden Pixel werden alle möglichen Geraden, die diesen Pixel enthalten, im Hough-Raum eingetragen. Dafür müsste zunächst für jede Steigung zwischen $-\infty$ und $+\infty$ der entsprechende Y-Achsenabschnitt berechnet werden. Die Zahl der Steigungen, die tatsächlich zu behandeln ist, wird durch die Auflösung des Hough-Raumes begrenzt.

Da für die Auflösung in der Regel eine Zweierpotenz gewählt wird, beschränkt sich der erste Unterraum tatsächlich auf das Intervall [-1, 1[, schließt also die Steigung Eins aus. Dies geschieht, da es sich um eine gerade Anzahl an Schritten handelt, welche die Null mit einschließen.
Es ist davon auszugehen, dass bei einer üblichen Auflösung des Hough-Raumes von beispielsweise 512, die Steigung 255/256 hinreichend genau ist um den Bereich der Steigung 1 zu repräsentieren.

In einer `for`-Schleife wird jede Steigung $0 \leq$ `a` $<$ *Auflösung* des Hough-Raumes ausgewertet.

```
double slope = 2 * a * dSteps - 1;                                    (1)
double yintc = (double) – yscaled – xscaled * slope;                  (2)
b = (int) Math.round( (yintc + 1.0) / (2.0 * dSteps) );               (3)
```

Zeile (1) rechnet die Bildkoordinate `a` der Steigung in die kartesische Koordinate `slope` im Intervall [-1, 1[um.
Zeile (2) berechnet die kartesische Koordinate `yintc` des Y-Achsenabschnittes, wobei die Bildkoordinaten(`u`, `v`) des Pixels zunächst auch in skalierte, kartesische Koordinaten (`xscaled`, `yscaled`) umgewandelt wurden.
In Zeile (3) wird aus der kartesischen Koordinate `yintc` schließlich die Bildkoordinate `b` des Y-Achsenabschnittes.

Jetzt kann das Bildkoordinatenpaar (`a`, `b`) im Hough-Raum eingetragen werden.
Da zunächst alle Steigungen im Intervall [-1, 1[berechnet wurden, ergibt sich für (`a`, `b`) nur der erste oder der dritte Unterraum, da der zweiten Unterraum nur für Steigungen > 1 und < -1 gilt. Wie in 4.1.3 beschrieben, dürften jedoch keine Einträge für den dritten Unterraum berechnet werden.

Für die Steigungen der Intervalle]-∞, -1[und]1, ∞[wird in jedem Schritt der `for`-Schleife nun zu jeder Steigung `a` der Kehrwert gebildet und mit diesem der zugehörige Y-Achsenabschnitt berechnet.

```
double slope2 = 1.0 / (double) slope;
```

Diesmal ergeben sich theoretisch Einträge für den zweiten und dritten Unterraum, praktisch werden hier alle (`a`, `b`) für den zweiten Unterraum berechnet.

Bevor (`a`, `b`) in den zweiten Unterraum eingetragen werden können, müssen sie noch auf das Intervall [1/*a*, *b*/*a*[umgerechnet werden. Dies geschieht in den folgenden beiden Zeilen:

```
int a2 = (int) Math.round( (1.0 / slope2 + 1.0) / (2 * dSteps));
int b2 = (int) Math.round( (yintc2 / slope2 +1.0) / (2 * dSteps));
```

Ein Problem ergibt sich, wenn die Steigung `slope` gleich Null ist. In diesem Fall kann für `slope2` keine Steigung berechnet werden, da die Division durch Null nicht möglich ist.
Daher wird jeder letzte Wert `b2` des Y-Achsenabschnittes in einer Variable `btemp` abgelegt. Diese kann im Fall, dass die Steigung `slope2` und somit der Y-Achsenabschnitt `yintc` ebenfalls nicht berechnet werden können, abgerufen werden. Auf diese Weise wird der Eintrag für `b2` an der Stelle `a` für den Kehrwert der Steigung Null interpoliert.

```
if (slope == 0.0) { b2 = btemp; }
```

5.2.2 Die Methoden doPixelSub1( ) und doPixelSub2( )

Die Methoden `doPixelSub1()` und `doPixelSub2()` sind der wesentliche Bestandteil der Klasse `Cascaded2HT` und bauen auf der Methode `doPixel()` aus 5.2.1 auf.

Auf die zu prozessierenden Pixel des ersten Unterraums wird `doPixelSub1()` angewendet, auf die des zweiten Unterraums folglich `doPixelSub2()`. Da im dritten Unterraum keine zu prozessierenden Pixel vorhanden sind (siehe 4.1.3), ist eine dritte Methode `doPixelSub3()` für dieses Plug-In nicht nötig und wurde daher nicht implementiert.

Das ursprüngliche Bild entspricht mit seinen Koordinaten (x, y) einem ersten Unterraum (a, b). Da die Pixel des Kantenbildes in `doPixel()` dazu passend prozessiert wurden, entspricht diese der Methode für den ersten Unterraum `doPixelSub1()`. Von ihrem Namen abgesehen sind die beiden Methoden identisch (siehe daher 5.2.1).
Diesmal können jedoch auch Eintragungen im dritten Unterraum vorgenommen werden.

Für die Methode `doPixelSub2()` müssen die Bildkoordinaten in der Art auf kartesische Koordinaten umgerechnet werden, dass die Berechnung von Steigung und Y-Achsenabschnitt auf Basis der Koordinaten (a, b) und nicht ($1/a$, b/a) geschieht (siehe 4.3).

Dabei ergibt sich als Parametergleichung für alle Steigungen von -1 bis 1

$$a + bx + y = 0 \quad \text{(siehe 4.3).}$$

Bei der ersten Hough-Transformation lautete die Parametergleichung

$$ax + b + y = 0 \quad \text{(siehe 3.2).}$$

Die Parameter *a* und *b* können somit beim Eintragen in den ersten Unterraum ausgetauscht werden:

```
int a1 = b;
int b1 = a;
```

Da keine Maxima in dem Bereich detektiert werden, in dem `yintc` > `slope` und `yintc2` > `slope2` sein könnte, bleibt der Fall, dass die Steigung im Intervall]-∞, -1[bzw.]1, ∞[liegt und `slope2` ≥ `yintc2` ist. Für diesen Fall gilt die Parametergleichung

$$\frac{1}{a}y + \frac{b}{a} + x = 0 \Leftrightarrow ax + b + y = 0\,.$$

Sie entspricht also wieder der Parametergleichung der ersten Hough-Transformation $ax + b + y = 0$. Das Ergebnis wird in den dritten Unterraum eingetragen.

5.2.3 Die Methode collectVanishingPoints( )

Diese Methode sammelt die kartesischen Koordinaten der Fluchtpunkte in einem zweidimensionalen Array `vpoints[][]`. Sie geht dafür jeden Pixel ihres Eingangsbildes durch und rechnet die Bildkoordinaten aller weißen Punkte in kartesische Koordinaten um. Die Formeln für diese Berechnung sind unter 4.4.1 beschrieben.
Die x-Koordinate des `n`-ten Fluchtpunktes wird unter `vpoints[0][n]` eingetragen, seine y-Koordinate wird dementsprechend unter `vpoints[1][n]` eingetragen.
Außerdem wird jeder Fluchtpunkt mit `n+1` nummeriert in einem ImageJ-log-Fenster ausgegeben.

5.2.4 Die Methode calculateFocalLength( )

Die Methode `calculateFocalLength()` berechnet schließlich die Brennweite des Systems. Da die Koordinaten bereits in kartesischer Form vorliegen, müssen keine Umrechnungen zwischen Bildkoordinatensystem und kartesischem Koordinatensystem mehr stattfinden.

In drei `for`-Schleifen mit `p`, `q` und `r` werden alle Kombinationen möglicher Fluchtpunkttripel ausgewertet. Die Anzahl dieser Fluchtpunkttripel entspricht dem Binomialkoeffizienten „`vpmax` über 3“, wobei `vpmax` die Anzahl der zu detektierenden Fluchtpunkte ist.

Die Berechnung des Hauptpunktes H(`xH`, `yH`) als Höhenschnittpunkt des Fluchtpunktdreiecks erfolgt wie unter 2.2.1 beschrieben.

Liegen die Koordinaten des berechneten Hauptpunktes nicht in der Nähe des Bildzentrums, wird zum nächsten Fluchtpunkttripel übergegangen. Anderenfalls wird im folgenden Schritt die Brennweite berechnet und in einem zweidimensionalen Array `focalL[][]` abgelegt. So werden viele Kombinationen, die nicht den drei richtigen Fluchtpunkten entsprechen, bereits im Vorfeld automatisch aussortiert.
Als Nähe des Bildzentrums wurde der Bereich]-0,2|0,2[in x- und y-Richtung gewählt.

Die Brennweite `focalL[0][z]` des `z`-ten Fluchtpunkttripels wird aus dem Hauptpunkt H(`xH`, `yH`) und den Fluchtpunkten B(`xB`, `yB`) und C(`xC`, `yC`) berechnet. Da $f_{ab} = f_{ac} = f_{bc}$ ist, können die beiden Fluchtpunkte, welche an dieser Stelle zur Berechnung herangezogen werden, willkürlich gewählt werden.

Sofern Angaben über die Sensorbreite des Aufnahmegerätes gemacht wurden, `sensorwidth` folglich größer Null ist, wird zuletzt die effektive Brennweite in mm berechnet (siehe 4.4.3). Ein möglicher Cropfaktor wird mit einberechnet, wenn `crop` ungleich Eins ist.
Dabei ist `focalL[1][z]` die effektive Brennweite in mm zu `focalL[0][z]`.
Unter Angabe der Fluchtpunkte, die zur Berechnung der jeweiligen Brennweite herangezogen wurden, sowie den Koordinaten des zugehörigen optischen Zentrums, wird die Brennweite schließlich in einem ImageJ-log-Fenster ausgegeben.

5.3 Einstellbare Parameter

Innerhalb des Programmes gibt es einige Parameter, die der Benutzer verändern und für seine Zwecke anpassen kann. Diese werden im Folgenden kurz vorgestellt.

Die Standard-Parameter haben sich im Laufe der Erarbeitung dieser Bachelorarbeit als häufig zielführend herausgestellt.

`vpmax`	Anzahl der zu detektierenden Fluchtpunkte (mindestens 3) in Klasse `PluginApproximateFocalLength_`
`hough_res`	Auflösung der Parameterräume/Hough-Räume (Standard: 512) in Klasse `PluginApproximateFocalLength_`
`gsv`	Schwellwert, ab welchem Grauwert ein Pixel des Kantenbildes prozessiert wird (Standard: 210) in Klasse `Cascaded1HT`
`alpha`	Empfindlichkeit des Detektors (Standard: 0,01) in Klasse `HarrisCornerDetector`
`threshold`	Schwellwert für die „Ecken-Stärke“ (Standard: 2000) in Klasse `HarrisCornerDetector`
`dmin`	Abstand, den zwei Maxima mindestens haben (Standard: 20) in Klasse `HarrisCornerDetector`
`alphaVP`	Empfindlichkeit des Fluchtpunkt-Detektors (Standard: 0,005) in Klasse `HCDVanishingPoints`
`thresholdVP`	Schwellwert für die „Ecken-Stärke“ (Standard: 2000) in Klasse `HCDVanishingPoints`
`dminVP`	Abstand, den zwei Fluchtpunkte mind. haben (Standard: 150) in Klasse `HCDVanishingPoints`
`sensorwidth`	Sensorbreite des Aufnahmesystems in mm, sofern bekannt (sonst 0) in Klasse `CalculateFL`
`crop`	Cropfaktor, wenn das Bild in eine Richtung beschnitten wurde (sonst 1) in Klasse `CalculateFL`

6 Ergebnisse

Mit dem Plug-In können hinreichend gute Ergebnisse erzielt werden, sofern die auszuwertenden Bilder bestimmte Vorgaben erfüllen. Im Folgenden werden einige beispielhafte Auswertungen vorgestellt.

Alle Testaufnahmen in einer Übersicht sind im Anhang II „Verzeichnis der Aufnahmesysteme und Testaufnahmen“ zu finden. Da die Originaldateien zu speicheraufwendig waren, wurden die Bilder verkleinert. Um welchen Faktor dies geschah ist diesbezüglich nicht relevant, da die Kantenlänge in der Verarbeitung auf das Intervall [-1, 1] skaliert wird (siehe 2.2).

Die vorgestellten Ergebnisse sind, wenn nicht anders erwähnt, mit den Standard-Parametern (siehe 5.3) erzielt worden.

6.1 Rahmenbedingungen

Damit ein Foto mit diesem Plug-In zielführend ausgewertet werden kann, muss es einige Kriterien erfüllen:

- Es muss drei Fluchtrichtungen aufweisen. Die Bildebene der Kamera darf demnach im Moment der Aufnahme zu keiner Fläche in der realen Welt parallel sein.
- Das Objekt muss parallele und orthogonal zueinander stehende Geraden aufweisen. Besonders gut eignen sich Aufnahmen von Gebäuden.
- Es dürfen nicht zu viele störende Elemente in der Aufnahme vorkommen.

6.2 Die Testaufnahme [b008]

Die Testaufnahme [b008] ist der Fuß des mittleren Kranhauses im Kölner Rheinauhafen. Sie wurde mit der Canon EOS 40D [can40d] bei 13mm (den EXIF-Daten entnommen) Brennweite aufgenommen. Das Gebäude ist hell mit dunklen Querstreben und Fensterrahmen. Das Pflaster und die Bepflanzung werden bereits von der Hough-Transformation größtenteils ausgeschlossen, da ihre Kanten schwach sind und somit durch den Sobel-Operator kaum mit Grauwerten höher 210 detektiert werden.
Bei der Detektion der Maxima im ersten Hough-Raum werden vier Maxima detektiert, die nicht zu einem Fluchtpunkt gehören. Diese werden bei der zweiten Hough-Transformation

zwar mit ausgewertet, stören jedoch nicht bei der Detektion der Fluchtpunkte. Bereits die Bestimmung von drei Fluchtpunkten führt zu einem sinnvollen Ergebnis.

Abbildung 6.2

Oben links Testaufnahme [b008]. Oben rechts das ImageJ-log-Fenster mit den Koordinaten der Fluchtpunkte, des optischen Zentrums und der berechneten Brennweite. Die untere Reihe zeigt den Ablauf des Plug-Ins von dem Ergebnis der ersten Hough-Transformation über die detektierten Maxima sowie der Darstellung der zweiten Hough-Transformation bis hin zur Detektion der Fluchtpunkte. Diese Fluchtpunkte sind zur besseren Visualisierung in das Ergebnisbild der zweiten Hough-Transformation eingebettet

6.3 Weitere Testaufnahmen

Einige Testaufnahmen erzielen ähnlich gute Ergebnisse. Andere bedürfen der Detektion einer größeren Anzahl von Fluchtpunkten oder Änderungen der einstellbaren Parameter.

Die Bilder [b001] bis [b007] sind Aufnahmen des gleichen Gebäudes mit unterschiedlichen Aufnahmesystemen und Brennweiten. Bei den Aufnahmen [b010] und [b011] wurde das Bildformat von 3:2 auf 1:1 beschnitten. Das Bildzentrum hat sich dabei nicht geändert und störende Elemente am Bildrand wurden entfernt. Sie haben daher einen Crop-Faktor von 1,5.

6.3.1 Elf Standard-Auswertungen

An dieser Stelle wurden alle elf Testaufnahmen mit Hilfe der Standard-Parameter ausgewertet. Lediglich die zu detektierende Anzahl potentieller Fluchtpunkte wurde dabei angepasst, bis ein sinnvolles Ergebnis erzielt wurde. Aufnahmen, die bei zwölf Fluchtpunkten kein solches Ergebnis lieferten, wurden hier als nicht auswertbar eingestuft.

Aufnahme	Anzahl potentieller Fluchtpunkte	Brennweite laut EXIF-Daten	Berechnete Brennweite
[b001]	3	16mm	15,2mm
[b002]	4	12mm	12,1mm
[b003]	-	8mm	-
[b004]	-	14mm	-
[b005]	3	16mm	15,3mm
[b006]	4	18mm	17,2mm
[b007]	3	24mm	24,4mm
[b008]	3	13mm	13,2mm
[b009]	10	16mm	16,7mm
[b010]	3	14mm	14,4mm
[b011]	3	24mm	24,9mm

Tabelle 6.3.1

Übersicht der Auswertungen aller elf Testaufnahmen

Wertet man alle Bilder mit zwölf potentiellen Fluchtpunkten aus, wird lediglich in zwei Fällen, [b010] und [b011], eine weitere Brennweite ausgegeben. Diese kann jedoch jeweils ausgeschlossen werden, da sie keine sinnvollen Werte ergeben.

6.3.2 Änderung der Standard-Parameter

Um Bild [b003] und [b004] erfolgreich auszuwerten, können die einstellbaren Parameter geändert werden.

Für [b003] genügt es bereits, den Schwellwert `gsv` (Grauwert ab welchem ein Pixel des Kantenbildes prozessiert wird) auf 220 zu erhöhen. Dadurch werden die Kanten, welche die Hough-Transformation prozessiert, noch stärker gefiltert.
Schon bei drei zu bestimmenden Fluchtpunkten wird für das Bild eine Brennweite von 8,2mm (H(0,09|0,00)) ausgegeben. Aufgenommen wurde dieses Bild laut EXIF-Daten bei einer Brennweite von 8mm.

Bei [004] werden zu viele „Nebenmaxima“ im ersten Hough-Raum detektiert. Dies kann verhindert werden, in dem der Abstand einzelner Maxima `dmin` erhöht wird. Dadurch werden weniger Maxima ausgegeben. Das hat jedoch auch zur Folge, dass weniger Geraden zur Detektion der Fluchtpunkte zur Verfügung stehen.
In diesem Fall ergab sich für das Bild eine Brennweite von 14,5mm (H(-0,03|-0,02)). Laut EXIF-Daten wurde das Bild mit einer Brennweite von 14mm aufgenommen.

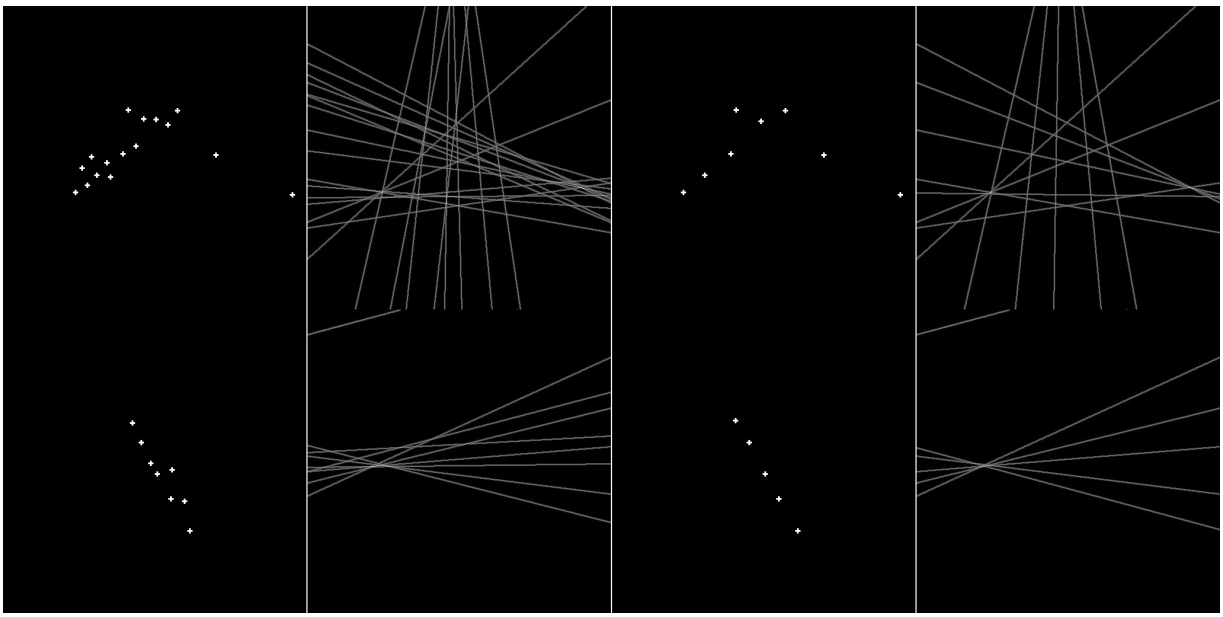


Abbildung 6.3.2a
Von links nach rechts: erster und zweiter Unterraum des Maxima-Bildes bei `dmin` = 20, erster und dritter Unterraum nach der zweiten Hough-Transformation bei `dmin` = 20 und die gleichen Unterräume für `dmin` = 40

Daraufhin den Standard-Wert für `dmin` auf 40 zu setzen und alle Aufnahmen auf diese Weise auszuwerten, bringt keine Verbesserung der Ergebnisse. Im Gegenteil, [b006] kann beispielsweise bereits bei `dmin` = 30 nicht mehr erfolgreich ausgewertet werden.
Zwar wird bei der Detektion einer größeren Anzahl von Fluchtpunkten wieder ein Ergebnis ausgegeben, dieses ist jedoch ungenauer, da die Fluchtpunkte schlechter detektiert wurden.

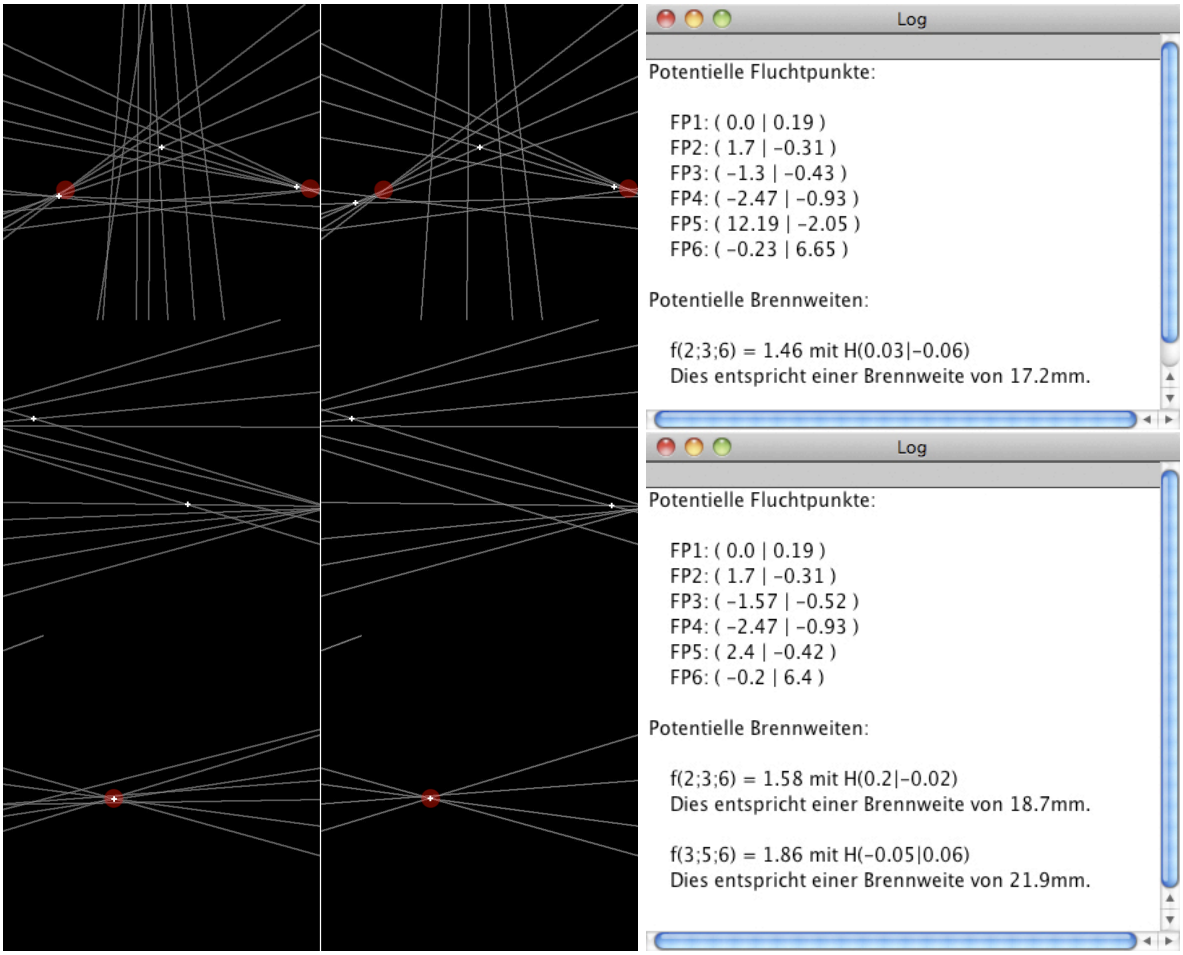


Abbildung 6.3.2b

Links ist der zweite Hough-Raum mit jeweils sechs detektierten Fluchtpunkten zu sehen. In der linken Spalte wurde `dmin` = 20 gewählt, in der rechten Spalte war `dmin` = 30. Die Bereiche, in denen die drei tatsächlichen Fluchtpunkte liegen, sind rot markiert. Oben rechts ist das Ergebnis für `dmin` = 20, unten rechts für `dmin` = 30

Die Parameter `dmin` als Mindestabstand der Maxima im ersten Hough-Raum sowie `dminVP` als Mindestabstand der Fluchtpunkte im zweiten Hough-Raum sollten anhand der Auflösung des Hough-Raumes gewählt werden. Verkleinert oder vergrößert man den Hough-Raum, ist es sinnvoll, `dmin` und `dminVP` mit dem gleichen oder einem ähnlichen Faktor zu multiplizieren.

6.3.3 Die erkannten Geraden im Bild und der Horizont

Das Bild [b010] zeigt den Fuß des Treppenhauses eines der Kranhäuser im Kölner Rheinauhafen. Da das Objekt durchscheinend ist, ist ebenfalls der Strand auf der gegenüberliegenden Rheinseite zu sehen. Dieser entspricht dem Horizont des Bildes und verläuft daher durch beide horizontalen Fluchtpunkte. Im ersten Unterraum des Hough-Raumes ist der Horizont der Schnittpunkt der beiden gedachten Geraden durch die Maxima.

Der Horizont ist nahezu parallel zur oberen und unteren Bildkante. Dies lässt sich auch an den ermittelten Koordinaten der Fluchtpunkte A(1,61|-0,76) und B(-2,47|-0,81) ausmachen, deren Y-Koordinaten beinahe auf einer Höhe liegen. Da der Horizont des Bildes nach unten verschoben ist (negative Werte für die Y-Koordinaten der beiden horizontalen Fluchtpunkte), handelt es sich bei der Aufnahme um eine Froschperspektive.

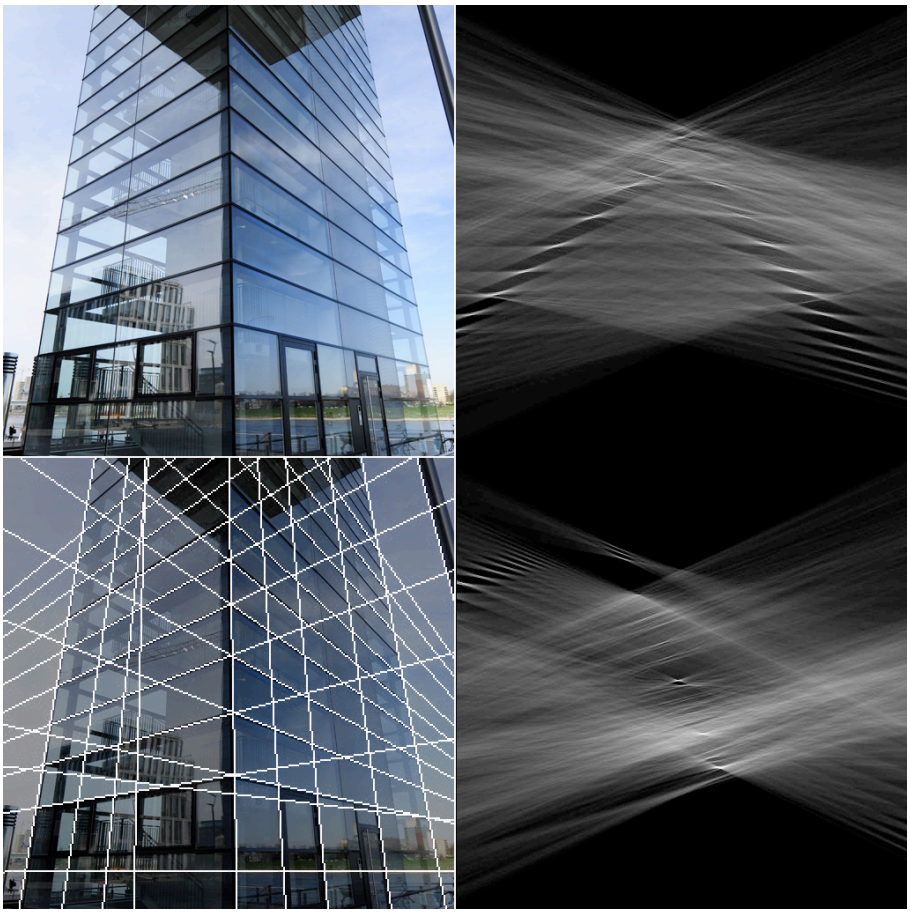


Abbildung 6.3.3

Links oben das Ausgangsbild, links unten die im Bild erkannten und durch die zweite Hough-Transformation ausgegebenen Geraden. Rechts der erste und zweite Unterraum der ersten Hough-Transformation. Die im Bild gefundenen Geraden sind sehr gut sichtbar und lassen bereits deutlich die drei Geraden mit den Parametern der Fluchtpunkte erkennen

7 Diskussion

Durch perspektivische Verzerrung schneiden sich in der Realität parallele Geraden in einer fotografischen Aufnahme in Fluchtpunkten. Sind letztere bestimmt, lassen sich mit ihrer Hilfe die inneren Kameraparameter optisches Zentrum und Brennweite berechnen. Zu untersuchen war, in wie weit sich diese Parameter automatisch bestimmen lassen. Am Ende sollte ein funktionierendes Java-Plug-In für das Open Source Bildverarbeitungsprogramm ImageJ stehen.

Zunächst bleibt festzuhalten, dass mit Hilfe des hier vorgestellten Plug-Ins das optische Zentrum und damit die Brennweite bestimmt werden können, wenn auch in engen Grenzen. Nach derzeitigem Stand müssen die zu detektierenden Linien gut ausgeprägt sein und störende Elemente nach möglichst gering gehalten werden. Wenngleich mit den einstellbaren Parametern (siehe 5.3) einige der Störfaktoren herausgefiltert werden können, können diese das Ergebnis nach wie vor beeinträchtigen.

Bei Aufnahmen, welche die notwendigen Voraussetzungen zur Auswertung mit diesem Plug-In erfüllen, können bereits hinreichend gute Ergebnisse erzielt werden.

Es bestehen jedoch Möglichkeiten den Programmablauf und das Ergebnis zu verbessern.

Bereits in der Vorbereitung des Bildmaterials kann eingegriffen werden. Sind auf einer Aufnahme viele Störelemente, könnten diese ausgeblendet werden, beispielsweise indem der Benutzer sie vor der Auswertung unschärfer macht. Das Plug-In könnte auch dahingehend erweitert werden, dass durch die manuelle Eingabe einer ROI (Bereich von Interesse) der zu analysierende Teil der Aufnahme eingegrenzt werden kann.

Weiteres Verbesserungspotential besteht bei der Kantendetektion. Der hier verwendete Sobel-Operator hat keine veränderbaren Parameter. Er ist daher nicht zu beeinflussen. Dabei wäre es von Nutzen, bereits an dieser Stelle einen Schwellwert angeben zu können, ab welcher Stärke eine Kante detektiert wird. Zudem wäre es hilfreich, wenn die gefunden Kanten schmaler, zum Beispiel als eine ein Pixel breite Linie, ausgegeben würden.

Der Harris Corner Detector genügte, um mit diesem Plug-In zu zeigen, dass das optische Zentrum und die Brennweite automatisch bestimmt werden können. Dennoch gibt es

leistungsfähigere Verfahren zur Detektion von Ecken, welche die Qualität der Ergebnisse noch einmal verbessern könnten.

An dieser Stelle wäre eine weitere Filterung hilfreich: Interessant zur späteren Bestimmung der Brennweite sind die gefundenen Maxima-Gruppen, die im Hough-Raum annähernd auf einer Geraden liegen und sich somit nach der zweiten Hough-Transformation in einem Fluchtpunkt schneiden. Liegt ein Maximum nicht mindestens mit zwei oder drei weiteren Maxima annähernd auf ihrer Regressionsgeraden, könnte es automatisch aussortiert werden. Zudem könnte ein Wert angegeben werden, wie weit die gefunden Maxima von dieser Regressionsgeraden abweichen dürfen, bevor sie verworfen werden.

Insgesamt wäre es ratsam, an diesen Punkten weiter zu arbeiten, da zur Berechnung des optischen Zentrums die genauen Koordinaten der Fluchtpunkte nötig sind, um ein exakteres Ergebnis zu erhalten.

Gleichzeitig gestaltet sich eine Aussage über die Genauigkeit der mit dem vorliegenden Plug-In erzielten Ergebnisse als schwierig. Die genutzten Arbeitsmittel lassen keine empirische Erhebung zu. Die mit den EXIF-Daten der Fotos gelieferten Brennweiten sind gerundet. Sind dort 13mm Brennweite angegeben, muss es nicht automatisch bedeuten, dass der Wert der tatsächlichen Brennweite entspricht. Die Aufnahme könnte beispielsweise auch bei 12,7mm oder 13,2mm Brennweite gemacht worden sein.

Wird also mit dem Plug-In eine Brennweite von 13,2mm errechnet, in den EXIF-Daten aber 13mm angegeben, muss das Ergebnis nicht falsch sein. Ebenso ist eine durch das Plug-In bestimmte Brennweite von 13,0mm nicht automatisch richtig, nur weil in den EXIF-Daten 13mm angegeben sind. Es bleibt daher festzuhalten, dass eine klare Aussage über die Genauigkeit der Ergebnisse bisher nicht möglich ist. An dieser Stelle müsste mit Bildern gearbeitet werden, deren exakte Brennweite bekannt ist, um präzise Vergleichswerte zu haben.

Für die Zukunft wäre auch noch die Bestimmung des Horizontes mit Hilfe des Plug-Ins möglich. Es müsste dafür eine dritte Hough-Transformation implementiert werden [3]. Der Horizont entspricht der Geraden, die durch die beiden horizontalen Fluchtpunkte verläuft. Er kann somit als Schnittpunkt der Geraden, welche die Fluchtpunkte durch eine weitere Hough-Transformation bilden, bestimmt werden.

Das Plug-In arbeitet innerhalb der gesetzten Vorgaben hinreichend genau, um sagen zu können, dass die Bestimmung der inneren Kameraparameter, Brennweite und optisches Zentrum, mit Hilfe der Hough-Transformation in kaskadierter Form möglich ist. Die Berechnungen könnten unter Einbezug der vorstehend genannten Punkte weiter verfeinert werden.

Anhang

I Quellenverzeichnis

[1] W. Burger, M. Burge: *Digitale Bildverarbeitung, Eine Einführung mit Java und ImageJ*
2. Auflage, Springer-Verlag Berlin Heidelberg (2006)

[2] W. Walcher: Praktikum der Physik
7. Auflage, B.G. Teubner Verlag/GWV Fachverlage GmbH, Wiesbaden (2006)

[3] T. Tuytelaars, M. Proesmans, L. Van Gool: *The Cascaded Hough Transform as Support for Grouping and Finding Vanishing Points and Lines*
Lecture Nores in Computer Sciences, LNCS, vol. 1315, Seite 278-289 (1997)

[4] T. Tuytelaars, M. Proesmans, L. Van Gool, T. Moons: *A Cascaded Hough Transform as an aid in aerial image interpretation*
International Conference on Computer Vision, ICCV98, Seite 67-72 (1998)

[5] Prof. Dr. W. Hoffmann, Universität Bielefeld, *Perspektive in der Mathematik: Bestimmung des Betrachtungspunktes*
online: http://www.math.uni-bielefeld.de/~hoffmann/proj/summary.html (04/2011)

[6] P. V. C. Hough: *Method and means for recognizing complex patterns*
US-Patent 3,069,654 (1962)

[p1] ImageJ
http://rsbweb.nih.gov/ij/

[p2] Eclipse IDE for Java Developers
http://www.eclipse.org

II Verzeichnis der Aufnahmesysteme und Testaufnahmen

Die Testaufnahmen wurden am 02.04.2011 im Rheinauhafen in Köln aufgenommen. Lediglich Bild 11 [b011] wurde bereits am 26.11.2010 dort aufgenommen und diente lange als Haupttestaufnahme.

Folgende Kamera-Objektiv-Kombinationen wurden für die Aufnahmen genutzt:

[can40d]	Canon EOS 40D mit Sigma 8-16mm f4,5-5,6 DC HSM Sensorbreite: 22,2mm
[nikd300]	Nikon D300 mit AF-S Nikkor 14-24mm f2,8 G ED Sensorbreite: 23,6mm

[b001] b001_16mm_can40d_nocrop.jpg

[b002] b002_12mm_can40d_nocrop.jpg

[b003] b003_8mm_can40d_nocrop.jpg

[b004] b004_16mm_nikd300_nocrop.jpg

[b005] b005_16mm_can40d_nocrop.jpg

[b006] b006_18mm_nikd300_nocrop.jpg

[b007] b007_24mm_ nikd300_nocrop.jpg

[b008] b008_13mm_can40d_nocrop.jpg

[b009] b009_16mm_can40d_nocrop.jpg

[b010] b014_16mm_nikd300_ crop1,5.jpg

[b011] b011_24mm_ nikd300_crop1,5.jpg

III Abbildungsverzeichnis

Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Overath, den 18. April 2011

Unterschrift
(Leonie Friederike Kirk)